AMENDMENTS TO THE SPECIFICATION

Please amend the first full paragraph on page 12 as follows:

Figure 5 is a flowchart illustrating an installation process according to one embodiment of the present invention. Assuming a continuous compiler and runtime monitor are distributed with the software package as indicated in Figure 2, the installation process first checks if a continuous compiler has already been installed 505. If no compiler has yet been installed 505, a compiler is then installed on the user's machine 510. Similarly, if no runtime monitor has yet been installed 515, a runtime monitor is installed on the user's machine 520. The intermediate representation of the software is copied to the target machine 525. Next, a profile database is build for the user's machine ~~525~~ 530. This initial profile database includes details of the hardware configuration of the users machine. The intermediate representation can then be compiled 535 using the profile database to generate a customized executable for the user's machine. The details of this compilation are substantially similar to those of the recompilation described with reference to figure 7 below.

Please amend the first full paragraph on page 13 as follows:

Figure 6 is a flowchart illustrating program execution processing according to one embodiment of the present invention. First, the executable version of the application program is started 605. While the executable is running, the process is sampled at a controlled interval 610. That is, profile data is collected. While the CPU is busy 615,

execution of the application program and profile collection continues. When the CPU becomes idle 615, binary level profiles and high level intermediate representation profiles are generated ~~625~~ 620.

Please amend the first full paragraph on page 16 as follows:

Figure 9 is a block diagram illustrating a compiler annotation according to one embodiment of the present invention. As indicated, the annotation ar action node 900 contains several pieces of data. First is a major ID 905. This major ID 905 is made up of information based on the source file, line and column number. The action ID 910 is a unique ID to capture a particular action on a particular instruction. The previous actions 915 and next actions 920 are pointers to the previous and next action nodes in the list. Actions 925 is an indication of the action that is represented by this node. For example, DELETED, CREATED, etc. Phase 930 refers to the compiler optimization phase in which the node was created. Feedback data 935 includes data collected during program execution. This data can include number of branches 945, load/store information 950, branch probability 955, cache miss indication 960, ~~etc.~~ prediction 965 RC execution 970, unrolling factor 975, and spec statements 980.

Please amend the last paragraph on page 17 continuing on page 18 as follows:

Figure 13 is a flow chart illustrating annotation merging according to one embodiment of the present invention. First, a new action node is created 1305. The major ID from one of the previous action nodes of the instructions being merged is copied to the new action node 1310. A new action number is assigned to the new node 1315. Next, the previous actions pointer of the new node is set to a list of action nodes of the instructions being merged. The new node is added to the next actions list of the previous nodes 1325 and the compiler phase in which the node was merged is marked 1330. Finally, the action is marked as MERGED 1335.

Please amend the second full paragraph on page 16, lines 16-18 as follows:

According to one embodiment ~~of the present invention~~, annotations ~~can be implemented as a~~ may variably relate to linked [[list]] lists of data structures. Handling of such linked lists is routine in the art [[but]] and is described with reference to figures 10-14 as it relates to annotations.